

Analyzing Scenery Images by Monotonic Tree

Yuqing Song and Aidong Zhang
Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY 14260

Abstract

Content-based image retrieval (CBIR) has been an active research area in the last ten years, and a variety of techniques have been developed. However, retrieving images on the basis of low-level features has proven unsatisfactory, and new techniques are needed to support high-level queries. Research efforts are needed to bridge the gap between high-level semantics and low-level features. In this paper, we present a novel approach to supporting semantics-based image retrieval. Our approach is based on the monotonic tree, a derivation of the contour tree for use with discrete data. The structural elements of an image are modeled as branches (or subtrees) of the monotonic tree. These structural elements are classified and clustered on the basis of such properties as color, spatial location, harshness and shape. Each cluster corresponds to some semantic feature. This scheme is applied to the analysis and retrieval of scenery images. Comparisons of experimental results of this approach with conventional techniques using low-level features, demonstrate the effectiveness of our approach.

Keywords: Content-based image retrieval, image feature extraction, annotation, semantics retrieval, monotonic tree

1 Introduction

With the enormous growth of image databases, there is an urgent need to build efficient and effective image retrieval systems. Content-based image retrieval (CBIR) offers a promising technology to address this need, and a variety of CBIR techniques have been developed. In particular, content-based image retrieval using low-level features such as color [43, 38, 27], texture [22, 37, 36, 42, 21], shape [44, 23, 13, 24, 25, 14, 10, 16, 48, 49, 35, 45, 34] and others [29, 39, 2, 15, 7] has been well studied. Various image-querying systems, including QBIC [11], VisualSeek [38], PhotoBook [28], Netra [19], and Virage [5], have been built, using low-level features for general or specific image-retrieval tasks.

However, retrieving images via low-level features has proven to be unsatisfactory. Effective and precise image retrieval by semantics still remains an open problem because of the extreme difficulty of fully characterizing images. Theories and approaches for querying and browsing images based on image semantics become increasingly critical as web-based information retrieval grows in popularity. In recognizing the existing problems in the CBIR field, we believe that research efforts are needed to bridge the gap between the high-level semantics of interest to users and the low-level features which can be efficiently extracted.

J.P. Eakins [9] classified image features into three levels, ranging from the highly concrete to the very abstract. For our purpose, we classify image features in three levels, as follows:

- *Primitive level*: Primitive features include color, texture, shape, and the spatial location of image elements.
- *Local semantic level*: Local semantic features describe the presence of individual objects in images. Two examples of queries by local semantic features are “find pictures with a bridge” (object of a given type) and “find pictures with sky and trees” (combination of objects).
- *Thematic level (or global semantic level)*: Thematic features describe the global meanings or topics of images. Two examples of queries at this level are “find pictures of a Chinese garden” and “find pictures of an earthquake.” The thematic features of an image are based on all objects in the image along with their spatial relationships. High-level reasoning is needed to derive the global meaning of all objects in the scene and to determine the topic of the image.

Most existing CBIR systems are based on primitive features [11, 28, 38, 19]. Although some approaches [12, 46, 4] have been proposed for image retrieval by semantic features, there is currently no CBIR system which works effectively with semantic features. The difficulty of detection and recognition of general objects presents a significant challenge to the development of a CBIR system which can extract general semantic features from images. However, techniques can be successfully applied to the simpler realm of specific domains.

In this paper, we focus on scenery images, a popular testbed for semantics extraction. For several reasons, scenery images are easier to analyze than general images. First, scenery images contain a limited range of object types. Common scenery object types include sky, tree, building, mountain, lawn, water, and snow. Second, shape features, which are difficult to characterize and match, are less important than other low-level features in analyzing scenery images. Finally, scenery images often fall into a small number of typical patterns, such as those shown in Figure 1. These typical patterns greatly simplify scene interpretation for scenery images.

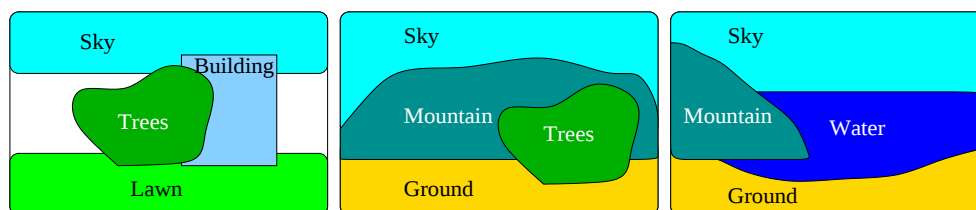


Figure 1: Some styles of scenery images.

In this paper, we will present a novel approach to supporting semantics-based image retrieval. This approach is based on the concept of the monotonic tree [40], a derivation of the contour tree for use with

discrete data. The structural elements of an image are modeled as branches (or subtrees) of the monotonic tree of the image. These structural elements are classified and clustered on the basis of such properties as color, spatial location, harshness, and shape. Each cluster corresponds to some semantic feature. Following these steps, images can be automatically annotated with category keywords, including sky, building, tree, wave, lawn, water, snow, and ground, thus facilitating high-level (semantics-based) image querying and browsing.

A scenery-querying system, termed *SceneryAnalyzer*, has been built to demonstrate the effectiveness of this approach. *SceneryAnalyzer* handles six types of scenery features: sky, building, tree, water wave, placid water, and ground. The visual properties of water wave and placid water are structurally different, and they are therefore treated as separate types. The ground feature in images can be further split into snow, lawn, and other subtypes.

The process of generating the semantic keywords which support high-level scenery querying is shown in Figure 2. Feature extraction is the central and unique component of this system. The extraction of features is accomplished off-line; each image is processed to extract semantic features, which are then stored in a feature vector. These semantic features are automatically annotated onto a gray copy of the original image. All feature vectors are stored in a feature base.

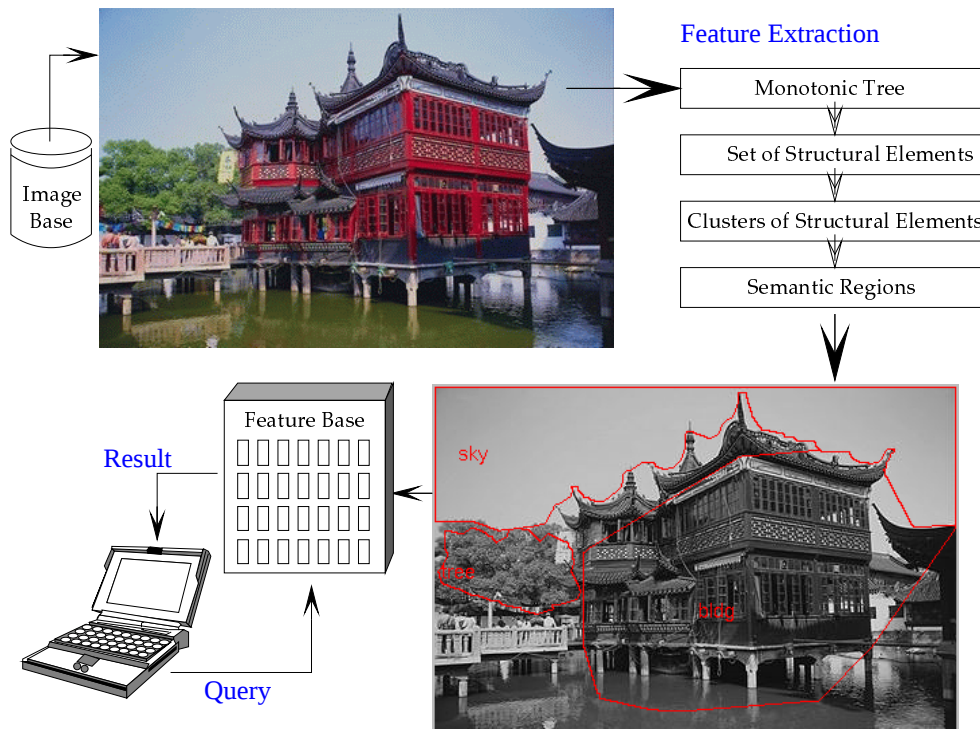


Figure 2: *SceneryAnalyzer* architecture.

The body of this paper will be devoted to a presentation of the details of the approach to semantic

feature extraction, including experimental results. Section 2 introduces the concept of the monotonic tree. In Section 3, techniques for extracting semantic features are described. Section 4 presents case studies of scenery features, while Section 5 offers a performance evaluation of the proposed approach. A summary and concluding remarks appear in Section 6.

2 Monotonic tree

2.1 Introduction to the monotonic tree

Contour trees [26, 33, 47] have been used in geographic information systems (GIS) and medical imaging to display scalar data. For example, elevation in the landscape can be modeled by scalar data over the plane, where a contour (also called an isoline) is a line along which the elevation function assumes a constant value. Contours are defined only for continuous functions. For an image represented by discrete data, a continuous function is first defined through interpolation of the data. The contour tree is then defined on this continuous function.

In [40], we introduced a new concept, the *monotonic line*, which is defined directly on discrete data. An *outward-falling/climbing monotonic line* for a gray image is a boundary within the image which is characterized by higher/lower pixel values just inside the boundary than just outside. All monotonic lines in an image form a rooted tree, called a *monotonic tree*. A maximal sequence of uniquely-enclosing monotonic lines is called a *monotonic slope*. All monotonic slopes in an image form a *topological monotonic tree*. A monotonic slope is called *outward-falling/climbing* if all its monotonic lines are outward-falling/climbing. Figure 3 illustrates these concepts. For a color image, the monotonic tree and topological monotonic tree are constructed on its gray copy.

2.2 Reduced monotonic tree

The process presented in this paper employs a *reduced monotonic tree*, rather than the topological monotonic tree, because the reduced monotonic tree can be constructed recursively. In the monotonic tree of a gray image, the leaf nodes correspond to the boundaries of the local maximum/minimum regions in the image. Starting from the leaf nodes in the monotonic tree, we find the monotonic slopes which contain these leaf nodes. These monotonic slopes are the leaf nodes in the topological monotonic tree, which are called the *slopes at level one*. By removing these monotonic slopes, we produce a smoothed gray image. We then apply the same process to the smoothed image to generate the *slopes at level two*. The process is repeated recursively to produce a hierarchy of slopes. Together, these slopes form a *reduced monotonic tree*.¹

¹The parent-child relation in the reduced monotonic tree is defined in a similar manner to that for the topological monotonic tree.

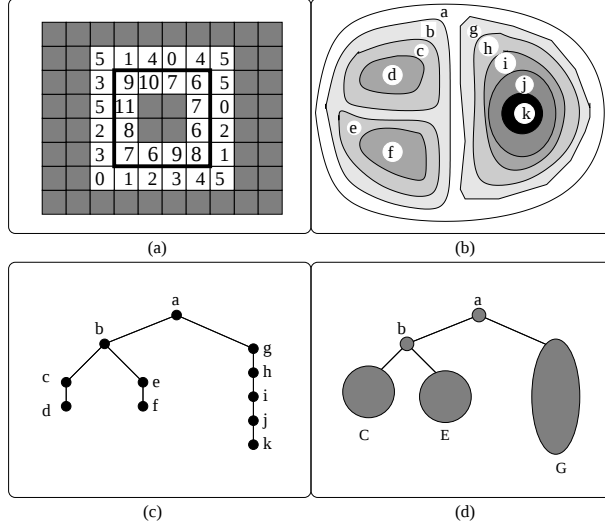


Figure 3: (a) An outward-falling monotonic line (the solid line in the figure), (b) a set of monotonic lines, (c) the monotonic tree, and (d) the topological monotonic tree.

Figures 4, 5, and 6 illustrate an algorithm for the generation of a reduced monotonic tree. The algorithm consists of the following steps:

- (1) Transfer the image to a hexagonal grid² (as shown in Figure 4). Denote the image on the hexagonal grid as $I = (f, \Omega)$, where Ω is the domain of the image and f is a function from Ω to $\mathbb{R}$. Let f_E be the extended function of f to the whole plane, which is equal to f on Ω , and assumes $-\infty$ out of Ω .
- (2) Find simply-connected regions³ where f_E assumes local minimum or maximum values, as shown in Figure 5(a).
- (3) For each local minimum/maximum, simply-connected region X , find the slope including ∂X ,⁴ where a slope is a sequence of monotonic lines with each line uniquely and directly enclosing the next, as shown in Figure 5(b). The procedure for finding the slope including ∂X is as follows:
 - (a) let $Y = X$ and $S = \{\partial Y\}$;
 - (b) if $Y = \Omega$, output S and stop;
 - (c) if ∂Y is outward-falling/climbing, let v be the highest/lowest value assumed by pixels⁵ adjacent to Y ;
 - (d) let $\{Z_i\}_{i=1}^n$ be the set of connected regions such that (i) each Z_i is adjacent to Y , and (ii) f_E assumes the constant value v over each Z_i . Let $Z = \bigcup_{i=1}^n Z_i$;
 - (e) if $\partial(Y \cup Z)$ is not a monotonic line, output S and stop;
 - (f) let $Y = Y \cup Z$ and $S = S \cup \{\partial Y\}$; go to step (b).

²Our theoretical definition of monotonic tree is based on a hexagonal grid.

³A simply-connected region is a connected region whose boundary is also connected. A finite connected region is simply-connected iff it has no hole.

⁴ ∂X is the boundary of X .

⁵Here, by “a pixel assumes value v ” we mean that f_E assumes v over this pixel.

All these slopes are nodes of the reduced tree, as shown in Figure 6(a).

- (4) If only one slope is found in step (3) and it covers the whole image, stop. Otherwise, smooth the image by removing all the slopes found in step (3). Let $\{X_i\}_{i=1}^n$ be the set of regions covered by those slopes, respectively. For each X_i , let v_i be the highest/lowest value assumed by pixels adjacent to X_i if ∂X_i is outward-falling/climbing. The smoothed image is a function $\tilde{f} : \Omega \rightarrow \mathbb{R}$ such that:

$$\tilde{f}(x) = \begin{cases} v_i & \text{if } x \in X_i \text{ for some } i; \\ f(x) & \text{otherwise.} \end{cases}$$

Figure 6(b) shows the smoothed image for our example. Let $f = \tilde{f}$ and go to step (2).

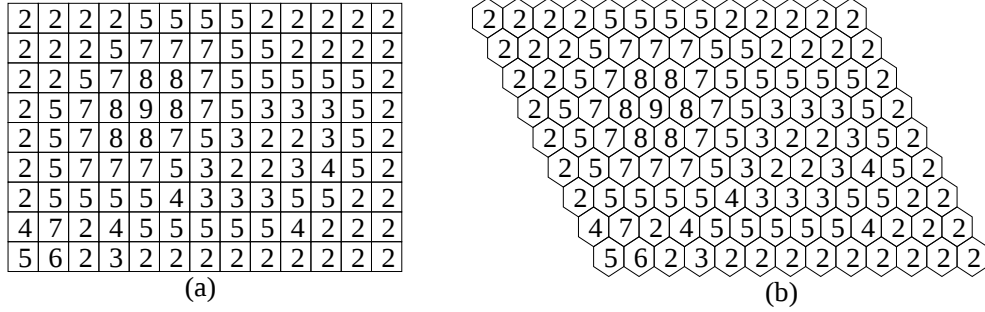


Figure 4: (a) A gray image with a square grid (each integer represents a pixel value in the image), (b) the gray image transferred to a hexagonal grid.

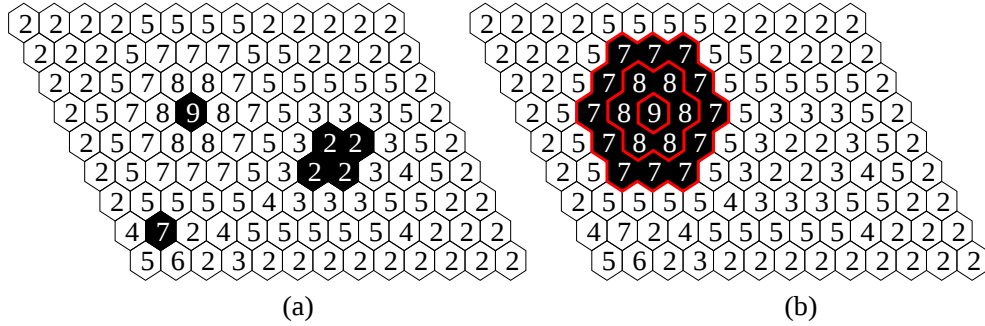


Figure 5: (a) Simply-connected regions where function f_E assumes local minimum/maximum values, (b) a slope with respect to a local maximum region (the red lines are the monotonic lines in the slope).

After the reduced monotonic tree is generated, it's translated back to a square grid. For the example image given in Figure 7 (a), its reduced monotonic tree is represented by its elements (nodes) at levels from 1 to 7, which are visually represented in Figure 7 (b)-(h), respectively. In these pictures, an outward-falling/climbing monotonic slope is shown by the white/black area which is enclosed by this slope. The blank area is shown in gray. The root node of the reduced monotonic tree of this example is at level 8. The root node covers the whole image.

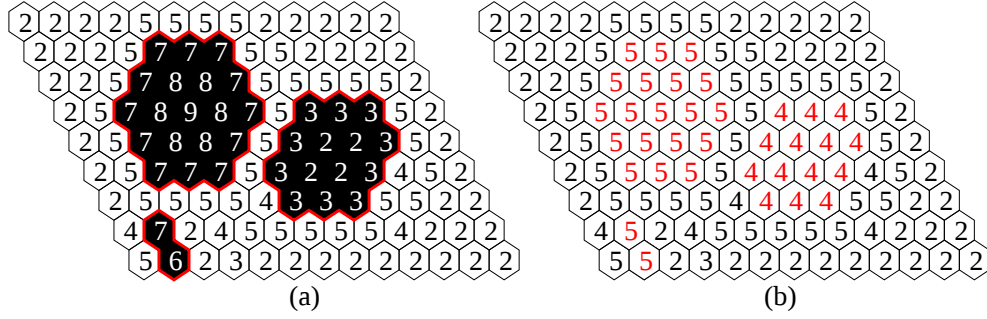


Figure 6: (a) All slopes with respect to local minimum/maximum regions, (b) the smoothed image.

Compared with other hierarchical or multiscale models, such as wavelet [20, 41], scale space [18], and peaks & ridges [6], the monotonic tree model has the following advantages:

- (1) The monotonic tree retrieves and represents the structures of an image at all scales. In addition, these structures are organized hierarchically as a tree, allowing easier analysis of the relationships between different levels.
- (2) The monotonic tree retrieves the structures of an image directly and maintains their original shapes.
- (3) Image structures are classified by their low-level features, then organized into clusters. Semantic features are drawn from the features of the clusters. The monotonic tree provides a vehicle for combining primitive features to characterize and capture semantic features.

3 Extraction of semantic features

The feature extraction scheme used here is based on the reduced monotonic tree. We use the branches of the reduced monotonic tree to model the basic structures in an image, which are termed *structural elements*. Structural elements have low-level features such as color, shape, and spatial location. They are clustered to form high-level features.

Feature extraction consists of three consecutive steps:

- (a) classifying structural elements;
- (b) clustering structural elements; and
- (c) rendering semantic regions.

Figure 8 shows the relationships among the steps given above.

The features of a given image are extracted and stored in a feature vector. A feature vector is an array which records the qualifying scores for all semantic features. The qualifying score of an image with respect

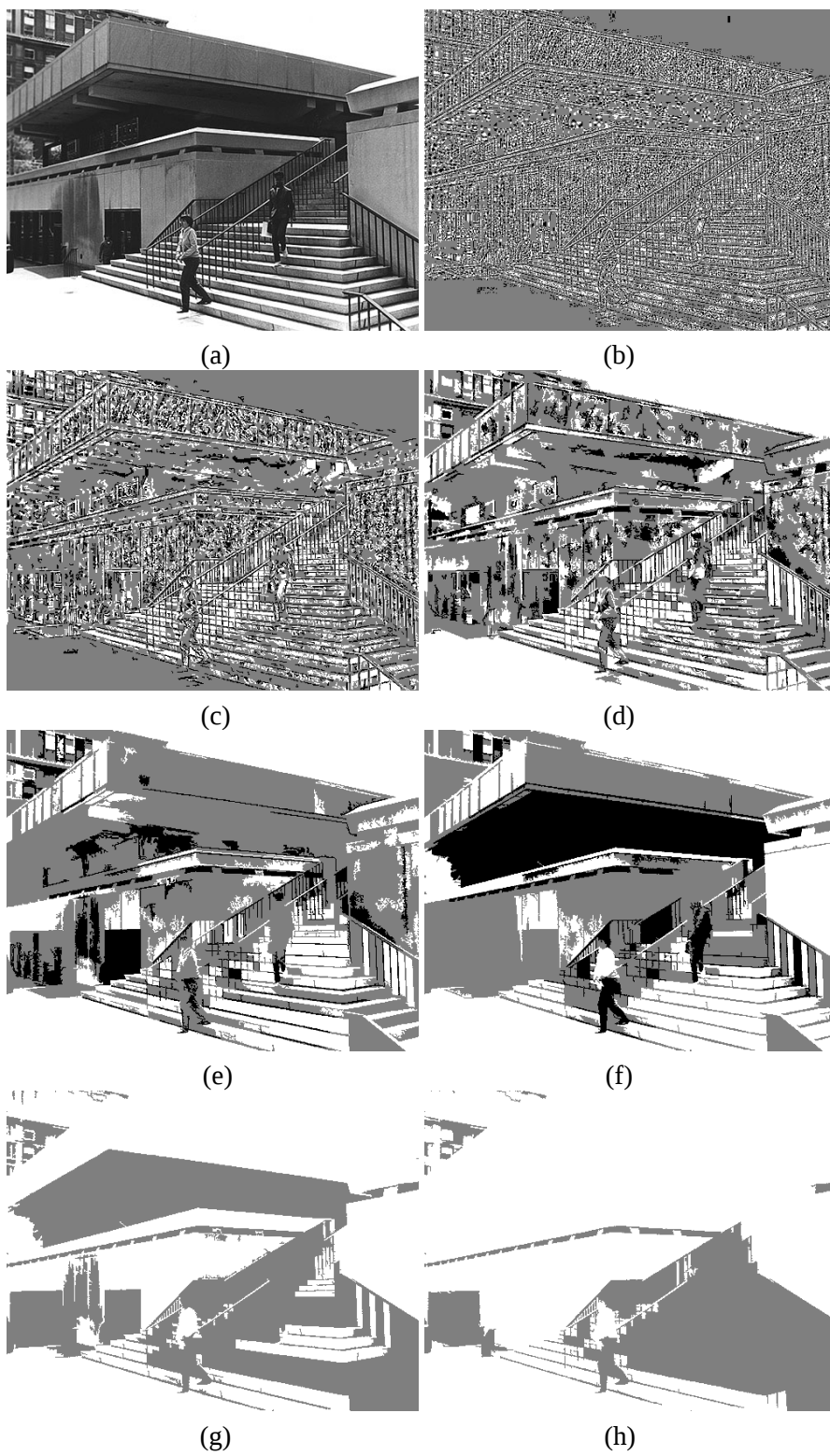


Figure 7: An example of reduced monotonic tree.

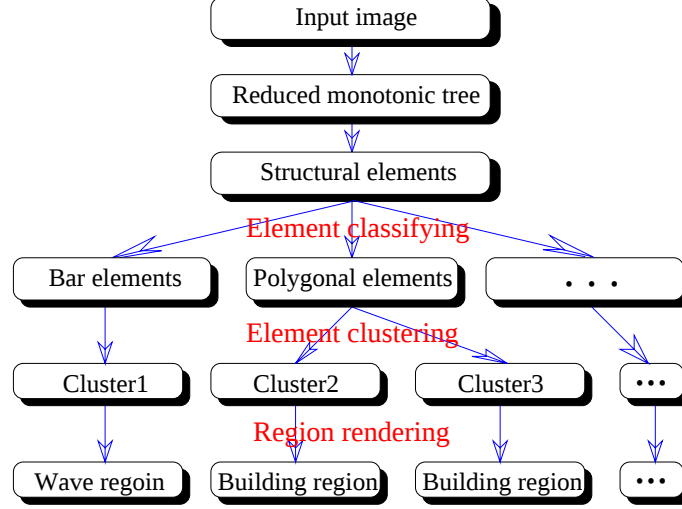


Figure 8: System design for feature extraction.

to a feature indicates the degree that the image can be considered to have this feature. Figure 9 illustrates the data structure of a feature vector.

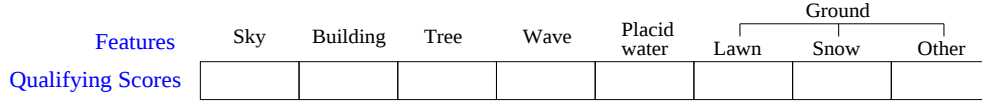


Figure 9: Data structure of feature vector.

3.1 Classifying structural elements

Each branch (or subtree) of a reduced monotonic tree is called a *structural element* if the image area it covers doesn't exceed an area threshold. A structural element is considered *positive/negative* if its root (i.e., the root of the subtree) is outward-falling/climbing, as shown in Figure 10(a). Positive/negative elements are like peaks/valleys. The *altitude* of a positive/negative element is defined as the absolute value of the average altitude of all its pixels above/below the highest/lowest pixels adjacent to the structural element, as shown in Figure 10(b). The *harshness* of a structural element is determined by the number, area, and altitude of its sub-elements.⁶ We define the harshness of an element t by

$$Harshness(t) = \frac{\sum_{b \in SubElementSet(t)} Altitude(b) * Area(b)}{Area(t)},$$

where $SubElementSet(t)$ is the set of sub-elements of t , $Altitude(b)$ is the altitude of b , and $Area(b)$ is the area of the region covered by b .

A structural element can be classified by its:

⁶An element is a branch of the reduced monotonic tree. All sub-branches of this branch are sub-elements.

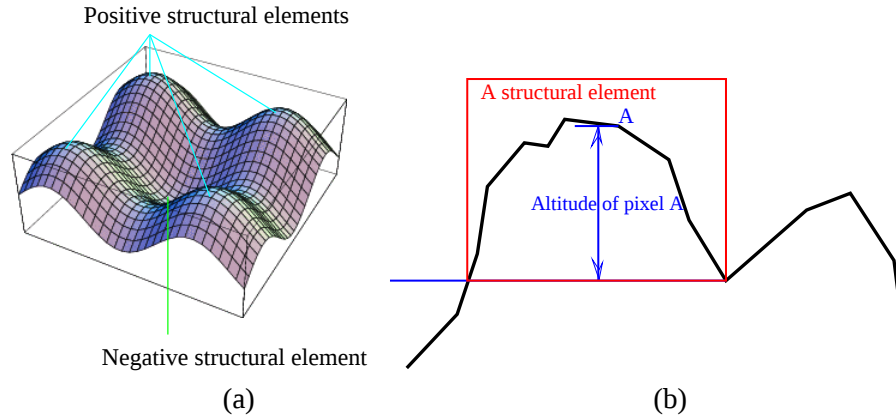


Figure 10: (a) Positive/negative structural elements; (b) the altitude of a pixel in a structural element.

- (1) color (the average color of all pixels in this element);
- (2) altitude;
- (3) harshness; and
- (4) shape (the shape of its covered region).

For example, we can classify the structural elements by shape as:

- (a) bars;
- (b) polygons;
- (c) irregular elements;
- (d) boundary-smooth elements; and
- (e) other shapes.

A bar element is characterized by its high length-to-width ratio, while a polygonal element is a structural element whose boundary consists mainly of line segments. A smooth-boundary element has a boundary which forms a smooth curve. Irregular elements are those with irregular boundaries. Figure 11 shows different types of structural elements.

The semantic features of scenery images are characterized by categories of structural elements which are clustered in the images. Three examples of the categories are polygonal elements (for buildings), horizontal bar elements (for waves), and green harsh irregular elements (for trees), as shown in Figures 12 and 13.

3.2 Clustering structural elements

Once the structural elements of an image have been classified, algorithms are applied to identify clusters of elements. First, the set of qualified (i.e., “belong-to”) elements must be determined for each category. If

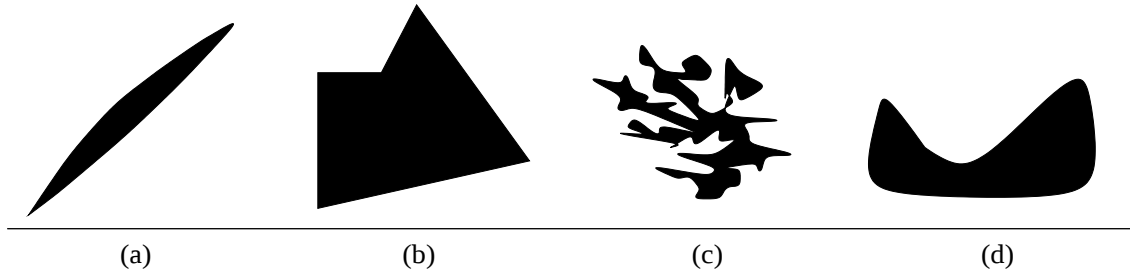


Figure 11: (a) A bar element, (b) a polygonal element, (c) an irregular element, and (d) a smooth-boundary element.

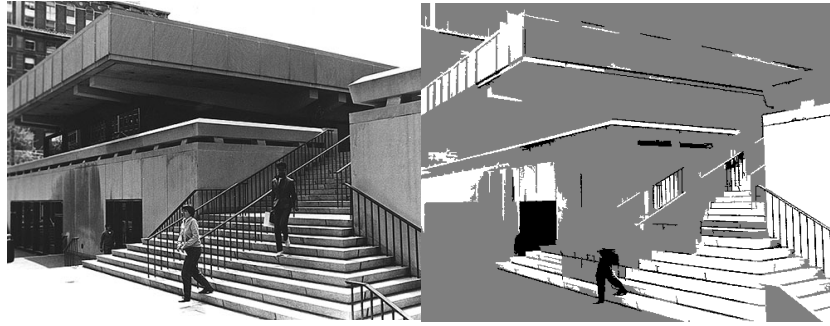


Figure 12: An image and its polygon elements (shown in black and white).

two qualified elements overlap, the one with the lower qualifying score⁷ is eliminated from the category. This process is called *element sifting*. The sifting process reduces the multi-level elements of the image to single-level elements, all belonging to the given category. The elements after sifting form an element pattern in the 2D plane.

Next, as shown in Figure 14, the Delaunay graph of the element pattern is constructed; this is the neighboring graph of the element pattern. We then apply clustering algorithms to the neighboring graph to find the clusters in the element pattern. Here, the clustering algorithm is based on the minimal spanning tree of the neighboring graph. Briefly described, the process of clustering by minimal spanning tree is as follows. Let V be the vertex set of the graph and D be the distance threshold. Then V is grouped into disjoint sets by joining all edges in the minimal spanning tree whose lengths (or weights) are less than or equal to D . Each set thus obtained is said to be a cluster at level D . Further background on pattern processing by neighboring graphs can be found in [1, 3]. References on clustering by minimal spanning tree can be found in [50, 8].

3.3 Rendering semantic regions

The process of rendering semantic regions from a cluster of structural elements consists of three steps: (1) element-connecting; (2) hole-filling; and (3) boundary-smoothing. In the first step, all elements in the

⁷For a given category, the qualifying score of an element indicates the degree that the element can be considered “qualified” to belong to this category. The process of arriving at a qualifying score will be discussed later.



Figure 13: An image and its green harsh irregular elements (shown in green or dark green) as well as its horizontal bar elements (shown in black and white).

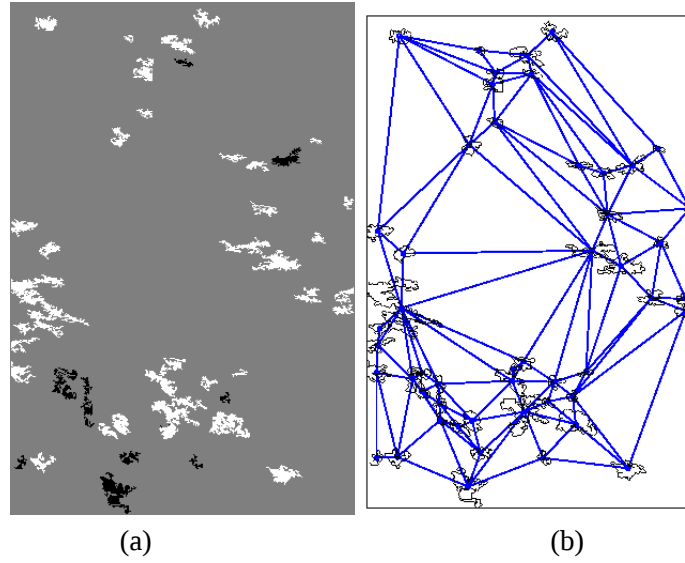


Figure 14: (a) An element pattern, and (b) the neighboring graph of the pattern.

cluster are connected by line segments of lengths within a threshold. The second step involves filling all holes which are smaller than a given area threshold. Finally, the boundary of the region is smoothed by removing irregular angles and branches. These steps are illustrated in Figure 15.

3.4 Qualifying scores

Qualifying scores must be established for structural elements, regions, and images. The qualifying score of a structural element measures the degree to which the element is considered to belong to a given category of elements. Qualifying scores for different element categories are determined in different ways. Formulas for computing the qualifying scores for various scenery-feature element categories will be presented in Section 4.

Similarly, the qualifying score of a region with a semantic feature measures the degree that the region manifests this particular feature. For a region generated from a cluster, its qualifying score is determined by

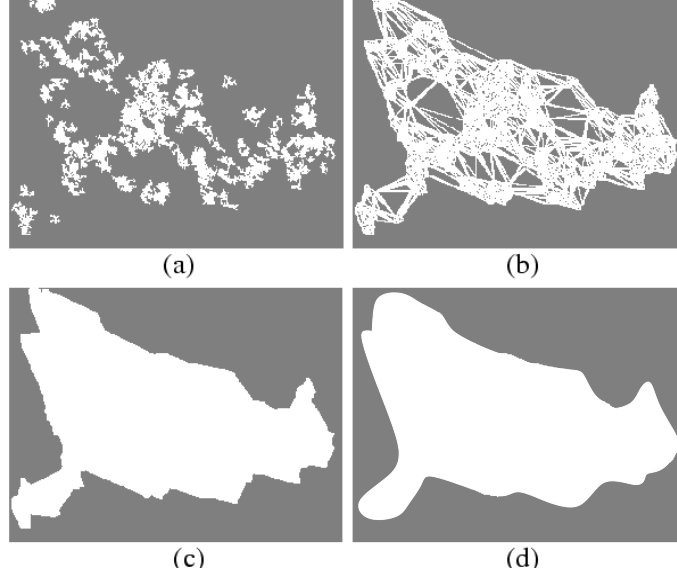


Figure 15: (a) A cluster of structural elements, (b) element connecting (c) hole filling, and (d) boundary smoothing.

the qualifying scores of the elements in the cluster and the area of the region. For example, a wave region is rendered from a cluster of horizontal bar elements. Let the input image be defined on the domain Ω and R be a wave region generated from a cluster S of horizontal bar elements. The qualifying score for R with respect to wave is defined as

$$Score_{wave}(R) = \frac{\sum_{t \in S} Score_{hbar}(t)}{\alpha_{wave}} * \frac{Area(R)}{Area(\Omega)},$$

where $Score_{hbar}(t)$ is the qualifying score for t to be a horizontal bar element, and α_{wave} is a parameter to scale the qualifying score. $Score_{hbar}(t)$ is defined in the next section.

In contrast, sky and ground regions are not generated from clusters. As will be discussed in the next section, we will assume that these regions are smooth. Coarse regions are identified first; the smooth regions are then the complement of the coarse regions. The qualifying score of a smooth region is determined by the smoothness, color, and area of the region. As with the harshness of a structural element, the smoothness of a region is determined by the area and altitude of the structural elements intersecting this region:

$$Smoothness(R) = \frac{Area(R)}{\sum_{t \in ElementSet} Altitude(t) * Area(Region(t) \cap R)},$$

where $ElementSet$ is the set of all structural elements of the input image and $Region(t)$ is the region covered by t . The qualifying score for a region R to be considered a sky region is defined as ⁸

$$Score_{sky}(R) = \frac{Smoothness(R)}{\alpha_{sky}} * \frac{Area(R)}{Area(\Omega)} * SkyColorRatio(R),$$

⁸Region R should first be examined to ascertain that it satisfies our assumptions about sky regions.

where α_{sky} is a parameter to scale the qualifying score and $SkyColorRatio(R)$ is the ratio of sky-colored pixels in R .⁹ The qualifying scores for ground and placid-water regions are defined similarly.

An input image may have several regions which manifest a specific feature. The qualifying score for the input image with respect to this feature is defined as the sum of qualifying scores of these regions. For example, let I be the input image and $R_1, R_2, \dots, R_n$ be the tree regions found in this image. The qualifying score of I with respect to “tree” is

$$Score_{tree}(I) = \sum_{i=1}^n Score_{tree}(R_i).$$

If an input image contains no regions which manifest a specific feature, then the qualifying score for the image with respect to this feature is zero.

4 Case study of semantic features

The previous section described a general scheme for the extraction of semantic features. In this section, we will present methods for defining the categories of structural elements and identifying the regions with scenery features.

The following discussion is predicated upon some simple assumptions about the colors of scenery features. For example, trees are assumed to be green and water to be blue. More comprehensive color patterns for scenery features can be integrated into our system, as is not the focus of this paper.

We will first briefly introduce our approach to color modeling. A pixel is a *gray* pixel if none of its red, green, and blue components is dominant; i.e.,

$$\max(r, g, b) - \min(r, g, b) \leq \mathcal{T}^g, \quad (1)$$

where r, g , and b are the red, green, and blue components of the pixel, respectively, and $\mathcal{T}^g$ is a threshold. A pixel is *red*, *green*, or *blue* if

$$r > \max(g, b) + \mathcal{T}^g, \text{ } g > \max(r, b) + \mathcal{T}^g, \text{ or } b > \max(r, g) + \mathcal{T}^g, \quad (2)$$

respectively. A pixel is *white* if it is gray and

$$(3 * r + 10 * g + b) / 10 \geq \mathcal{T}^w, \quad (3)$$

where $(3 * r + 10 * g + b) / 10$ is the brightness of the pixel and $\mathcal{T}^w$ is a threshold. A region is *red* (*green*, *blue*, *gray*, or *white*) if the ratio of red (green, blue, gray, or white) pixels in the region is greater than or equal to a threshold.

⁹We assume that the sky color is either blue or gray (the color of clouds). These assumptions will be discussed in the next section.

We also make assumptions about the location of sky and ground regions in images. For a region R in an image I , its location is modeled by its top, bottom, left, and right positions:¹⁰

$$TopPos(R) = \min\{y | \exists x, (x, y) \in R\}, \quad (4)$$

$$BottomPos(R) = \max\{y | \exists x, (x, y) \in R\}, \quad (5)$$

$$LeftPos(R) = \min\{x | \exists y, (x, y) \in R\}, \quad (6)$$

$$RightPos(R) = \max\{x | \exists y, (x, y) \in R\}. \quad (7)$$

Suppose the domain of image I is $\{0, 1, \dots, N - 1\} \times \{0, 1, \dots, M - 1\}$. We say that a region R is in the lower part of the image if $\frac{TopPos(R)}{M}$ is greater than a threshold; we say that a region occupies an upper part of the image if $\frac{TopPos(R)}{M}$ is less than a threshold.

4.1 Sky

The sky is a kind of background. Without clouds, a sky region is a homogeneous region consisting of blue pixels. Due to their physical properties, clouds in an image tend to change smoothly at the pixel level. From a location's stand point, there is usually no other object above a sky region. To retrieve sky regions, we make three simple assumptions:

- (sky.a1) A sky region is smooth;
- (sky.a2) A sky region occupies an upper part of the image; and
- (sky.a3) The color of sky regions is either blue or the color of clouds.

For our current implementation, we assume that the color of clouds is gray. To find the sky regions, we first identify the smooth regions in the image. The smooth regions are the complement of the harsh regions, which are characterized by peaks and valleys in intensity. Within the monotonic tree, these peaks and valleys are modeled as small structural elements with high altitudes. Thus, the harsh regions of the image are detected by clustering these small, high-altitude elements, as discussed in the last section. Identification of the harsh regions automatically distinguishes the smooth regions, as well. These smooth regions are then checked for location and color to find the sky regions.

4.2 Ground and placid water

We make three assumptions about ground regions:

- (ground.a1) A ground region is smooth;

¹⁰Here, we assume the monitor coordinate system: the x -axis is pointing right and the y -axis is pointing down.

- (ground.a2) A ground region is in the lower part of the image; and
- (ground.a3) In a ground region, structural elements are more horizontal than vertical.

When scenery images are taken, the direction of the projection is usually horizontal or nearly horizontal. Thus, as the natural scene is projected onto the image plane, ground-level structures appear more horizontal than vertical, leading to the third assumption above.

Let us first discuss methods for modeling the horizontality of a structural element. For a region X on the square grid $\mathbb{Z}^2$ and some $x_0 \in \mathbb{Z}$, we define the height of X at x_0 to be the number of pixels in the intersection between X and the vertical line crossing $(x_0, 0)$. Formally,

$$Height(X, x_0) = |\{(x_0, y) | (x_0, y) \in X\}|, \quad (8)$$

where $|\bullet|$ gives the number of elements in a set. Similarly, we define the width of X at y_0 as

$$Width(X, y_0) = |\{(x, y_0) | (x, y_0) \in X\}|. \quad (9)$$

Now we define the horizontal and vertical ratios of X to be

$$HorizontalRatio(X) = \sqrt{\frac{\sum_{y \in \mathbb{Z}} (Width(X, y))^2}{\sum_{x \in \mathbb{Z}} (Height(X, x))^2}}; \text{ and} \quad (10)$$

$$VerticalRatio(X) = \frac{1}{HorizontalRatio(X)}. \quad (11)$$

For a rectangle $\{0, 1, \dots, N-1\} \times \{0, 1, \dots, M-1\}$, its horizontal ratio is $\frac{N}{M}$. A structural element is said to be horizontal (or vertical), if the horizontal (or vertical) ratio of its covered region is greater than a given threshold.

As with the detection of sky regions, the identification of ground and placid-water regions starts with finding the smooth regions in the image. Then, for each smooth region, we check the validity of assumptions (ground.a2) and (ground.a3). To test (ground.a3), we count the horizontal and vertical elements in the smooth region. The last assumption holds if the horizontal elements exceed the vertical elements in the region.

A ground region identified by this method could be lawn, snow, or other ground subtypes. These region subtypes are distinguished by color. For example, we assume that lawn regions are green and snow regions are white.

Identification of placid-water regions is based upon four assumptions. The first three are the same as those used for ground identification. In addition, we assume that the color of placid water is blue.

4.3 Wave

Small water waves are characterized by very regular patterns which appear as horizontal bars in a horizontally-projected image, such as that shown in Figure 13. Breaking waves have complicated structures. However, images with breaking waves usually contain an area of water surface which consists of parallel bar structures, as in the image shown in Figure 16.

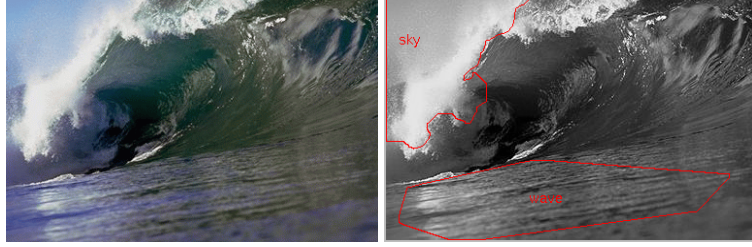


Figure 16: An example of an image with breaking waves and its annotation.

We define a wave region as a region consisting of horizontal bar elements. A bar is characterized by its high length-to-thickness ratio. For a 2D region X , we define its length and thickness as

$$Length(X) = \max_{x,y \in X} \|x - y\|, \quad (12)$$

$$Thickness(X) = \min_{v \in \mathbb{R}^2} \max_{x,y \in X} \frac{(x - y) \bullet v}{\|v\|}, \quad (13)$$

respectively, where $(x - y) \bullet v$ is the inner product of vectors $x - y$ and v , and $\|v\|$ is the length of the vector v . We define the bar ratio of a region X as

$$BarRatio(X) = \frac{Length(X)}{Thickness(X)}. \quad (14)$$

This situation is illustrated in Figure 17. A structural element is considered to be a bar element if the bar ratio of its covered region is higher than a given threshold. The qualifying score of a structural element to be a horizontal bar element is defined as the product of the bar ratio and horizontal ratio of its covered region:

$$Score_{hbar}(t) = BarRatio(Region(t)) * HorizontalRatio(Region(t)). \quad (15)$$

Wave regions are identified by clustering horizontal bar elements in the image.

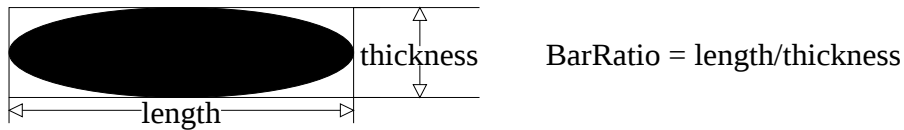


Figure 17: The bar ratio of an ellipse.

4.4 Green tree

A careful examination of the tree region in Figure 13 reveals that the micro-structures in the region are very irregular. Based on this observation, we assume that a tree region is a region consisting of green harsh irregular elements. The tree regions in an image are found by clustering the green harsh irregular elements in the image.

For a connected region X , let $ConvexHullLength(X)$ be the length of its convex hull and $Length(\partial X)$ be the length of its boundary. We define the irregularity of X by

$$Irregularity(X) = \frac{Length(\partial X)}{ConvexHullLength(X)}. \quad (16)$$

The qualifying score of a structural element to be a tree element is defined as the product of its harshness and the irregularity of its covered region:

$$Score_{treeElement}(t) = Harshness(t) * Irregularity(Region(t)). \quad (17)$$

4.5 Building

The shapes of most buildings are characterized by the line segments they contain. We assume that a building region in an image is a region consisting of polygonal elements. To ascertain whether a structural element is polygonal, we first partition its boundary into line segments and other kinds of segments. A structural element is polygonal if its boundary mainly consists of line segments. References on curve partitioning can be found in [30, 31, 32].

For a connected region X , let $Length(\partial X)$ be the length of its boundary and $LengthOfLineSegments(X)$ be the total length of the line segments detected in its boundary. We define the polygon ratio of X by

$$PolygonRatio(X) = \frac{LengthOfLineSegments(X)}{Length(\partial X)}. \quad (18)$$

The qualifying score of a structural element to be a polygonal element is defined as the polygon ratio of its covered region.

5 Experiments

Comprehensive experiments were conducted to demonstrate the effectiveness of the proposed approach; these experiments employed our SceneryAnalyzer system.¹¹ First, performance comparisons using COREL images were made between the proposed approach and some traditional CBIR techniques. These comparison measured the precision-recall for each scenery feature. We then conducted experiments using PhotoDisc

¹¹The system can be accessed at <http://monet.cse.buffalo.edu:8888/>.

<i>Feature</i>	sky	building	tree	wave	placid water	ground		
						lawn	snow	other
<i>COREL images with the feature</i>	2369	1920	1479	161	882	298	68	659
<i>PhotoDisc images with the feature</i>	455	156	382	6	219	57	59	261

Table 1: Breakdown by scenery features of images in COREL and PhotoDisc databases.

images to verify that the performance of our approach is not source-sensitive. Finally, the proposed approach was applied to both COREL and PhotoDisc images to explore its effectiveness with different combinations of scenery features. Examples of some of these images and their copies annotated by SceneryAnalyzer are provided at the end of this section.

5.1 Experiment setup

Two image databases were employed in these experiments. The first image database, COREL, consists of 6,776 color images from CD7 and CD8 of COREL Gallery 1,300,000. These photos are stored in JPEG format and are sized at either 256×384 or 384×256 . The COREL database includes both scenery and non-scenery images. There are 4,125 scenery pictures taken at globally diverse locations, along with 2,651 non-scenery images covering a wide variety of subjects, including fish, molecules, space scenes, insects, and other topics. The second image database, PhotoDisc, consists of 1,444 images from PhotoDisc Compiling Discs 3 and 4. These images fall into five categories: (1) homes and gardens, (2) international sports, (3) nature scenes, (4) panoramic landscapes, and (5) people, lifestyles, and vacations. Table 1 shows a breakdown of these two image databases by scenery features.

For each scenery feature, the precision-recall of SceneryAnalyzer was calculated and plotted. Let *RETRIEVELIST* be the list of n images retrieved with a given feature, which is sorted descendingly by qualifying scores. We calculate the precision and recall for each of the first $\frac{n}{30}$, $\frac{2n}{30}$, $\frac{3n}{30}$, ..., and n images in *RETRIEVELIST*. The precision and recall figures were then plotted to demonstrate the performance of SceneryAnalyzer.

In Subsection 5.2, performance with the COREL database will be used to compare SceneryAnalyzer with the keyblock model [17], traditional color histogram [43], color coherent vector [27], and wavelet (Haar and Daubechies) texture techniques [37, 41]. All these methods accept only queries by example. The process of selecting query sets and calculating the precision-recall for these methods can be illustrated with a typical case, where the sky feature is queried using the keyblock approach. There are 2,369 COREL images with sky regions. Each sky image is used as a keyblock query on the COREL database; the top 100 images are thus determined, and the number of sky images in this retrieved set is counted. The 2,369 sky images are then sorted in descending order by the number of sky images in their corresponding retrieved sets. The sorted list is denoted as *SKYLIST*. The first 5%, i.e., 118 of images from *SKYLIST* are then selected as the

query set and denoted as *QUERYSET*. For each COREL image I , we calculate its distance to *QUERYSET*- $\{I\}$ using the keyblock approach.¹² ¹³ The COREL images are then sorted in ascending order by this distance. The top 2,369 (i.e., the number of images with sky) COREL images are retrieved. Using the images retrieved by the keyblock model, we then calculate and plot the precision-recall of the keyblock model for the sky feature.

Subsection 5.4 compares the performance of SceneryAnalyzer with the traditional CBIR techniques on combinations of scenery features. For this comparison, these techniques are used to retrieve the top 20 images which have the specified combination of scenery features. Query sets are selected and images retrieved in the manner described above.

5.2 Comparison using COREL database

Experiments were conducted using 6,776 COREL images to compare SceneryAnalyzer with the keyblock approach [17], traditional color histogram [43], color coherent vector [27], and wavelet (Haar and Daubechies) texture techniques [37, 41]. The precision-recall of these techniques was compared for six scenery features: sky, building, tree, wave, ground, and placid water.

The precision-recall of each method was calculated for each scenery feature listed; these results appear in Figure 18. The five traditional CBIR methods all performed poorly with the wave feature. This can be partially explained by the small number of wave images in the COREL database, which makes these methods easily misidentify other images as wave images. Furthermore, these methods didn't capture the structures of waves.

A comparison of the graphs in Figure 18 shows that our method outperforms all the others for each scenery feature type. To further clarify the comparison, we calculated the average precision-recall over all six scenery features, which is shown in Figure 19(a). The average precision-recall of the five traditional methods suffers particularly from their poor performance with the wave feature, as discussed above. To exclude this effect, we also calculated the average precision-recall over the five remaining scenery feature types (sky, building, tree, ground, and placid water); this is shown in Figure 19(b). These comparisons indicate that our proposed method provides much better precision-recall of scenery features than the traditional techniques.

5.3 Experiments using PhotoDisc database

We also conducted experiments on 1,444 PhotoDisc images. The performance of SceneryAnalyzer with this PhotoDisc database is shown in Figure 20(a) and (b). Since the PhotoDisc database included only 6 images

¹²The distance from an image to a set of images is the shortest distance from this image to the images in the set.

¹³If we retrieve by the distances of images to *QUERYSET*, the images in *QUERYSET* will have distance zero. Thus the query images will automatically be retrieved as the top 5% images, which provides an unfair comparison.

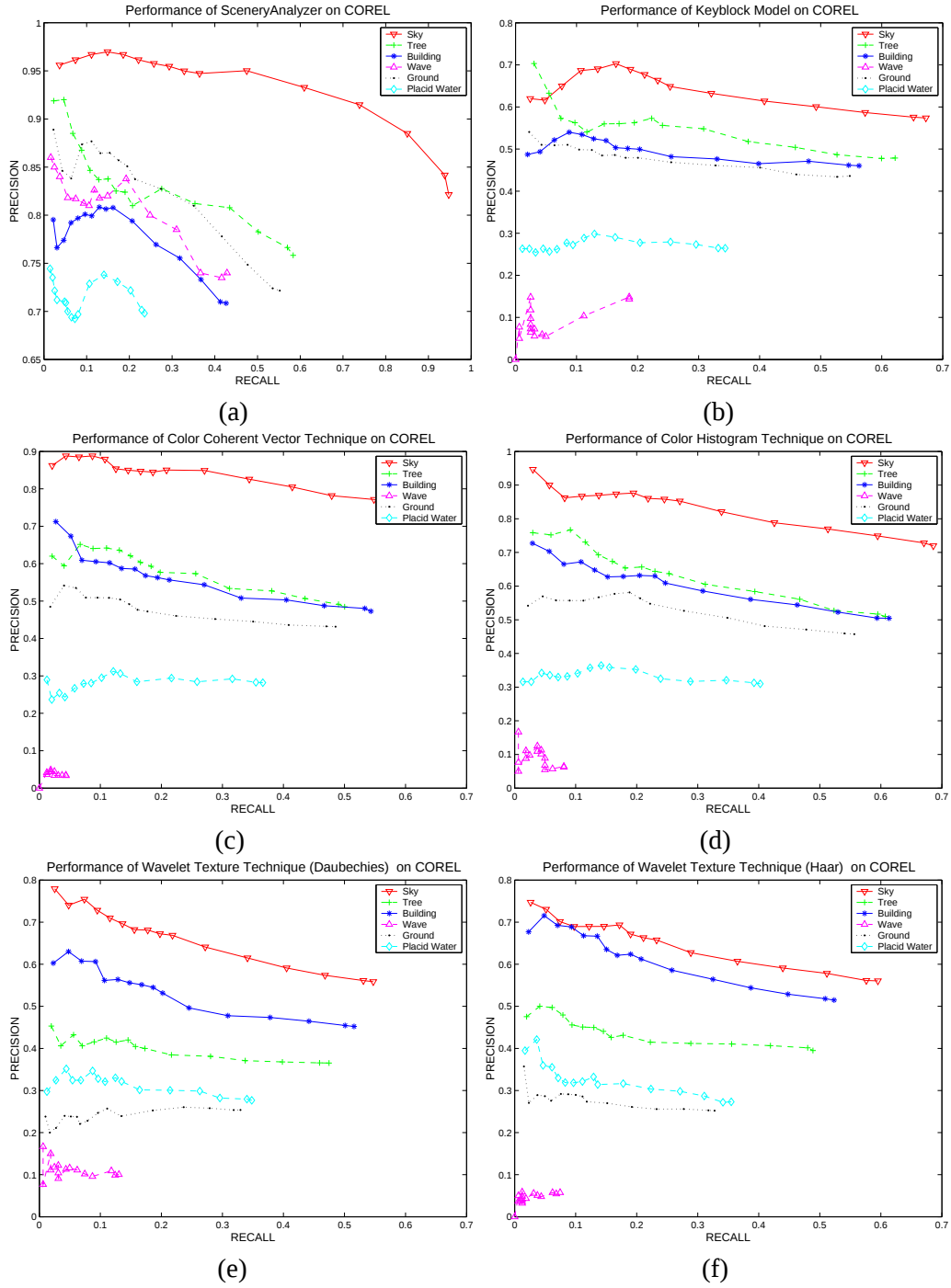


Figure 18: Performance of (a) SceneryAnalyzer, (b) keyblock model, (c) color coherent vector, (d) color histogram, (e) Daubechies wavelet, and (f) Haar wavelet with COREL images.

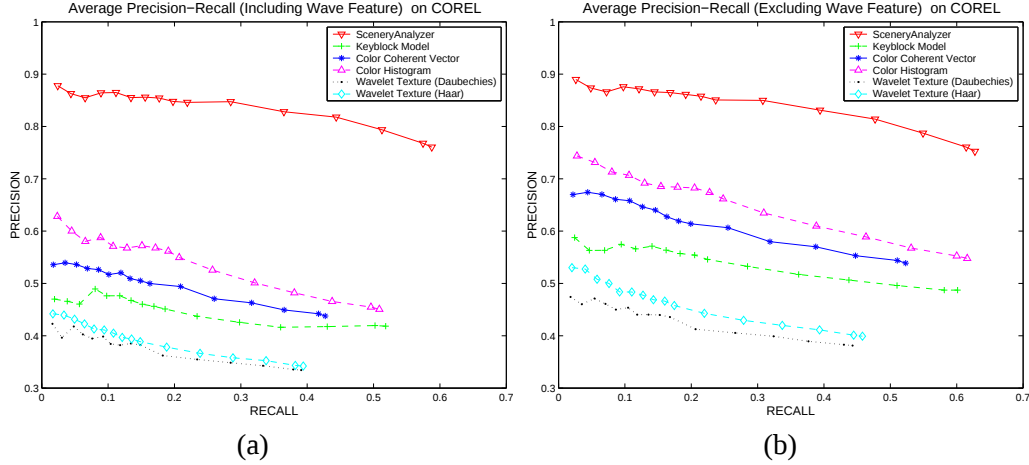


Figure 19: (a) Average precision-recall for six features: sky, building, tree, wave, ground, and placid water; (b) average precision-recall for five features: sky, building, tree, ground, and placid water.

with waves, a precision-recall curve could not be plotted for the wave feature. Thus, Figure 20(a) shows only five precision-recall curves. It's worth noting, however, that when queried for wave images on this database, SceneryAnalyzer retrieved three images, all with waves. Therefore, the precision of these queries was 100%, with a 50% recall. As a comparison, the performance of SceneryAnalyzer with the COREL database is shown in Figure 20(c) and (d). These experiments show that SceneryAnalyzer works effectively with both COREL and PhotoDisc images.

With both the COREL and PhotoDisc databases, the performance of SceneryAnalyzer is relatively poor with respect to snow subtype of ground. In our implementation, we have assumed that the ground is smooth and the snow ground is white. However, the snow ground in natural images may not be smooth and may vary in color. In addition, many other objects in natural images appear to be white and smooth. These factors negatively impact the performance with respect to snow ground.

5.4 Experiments on combinations of scenery features

Experiments were conducted to demonstrate the effectiveness of SceneryAnalyzer with combinations of scenery features. The experiments involved all the images in the COREL and PhotoDisc databases. The top 20 images for each combination of scenery features were retrieved and the precision and recall calculated; results are given in Table 2. In this table, for each combination of scenery features, $\mathbf{F}$ denotes the set of images with the stipulated features and $\mathbf{R}$ denotes the set of images retrieved. For a set X , $|X|$ denotes the number of elements in X . As a comparison, the performance of the five traditional CBIR techniques is given in Table 3, where the keyblock model is abbreviated as kb, the color histogram technique as ch, the color coherent vector as ccv, the Daubechies texture technique as daub, and the Haar texture technique as haar.

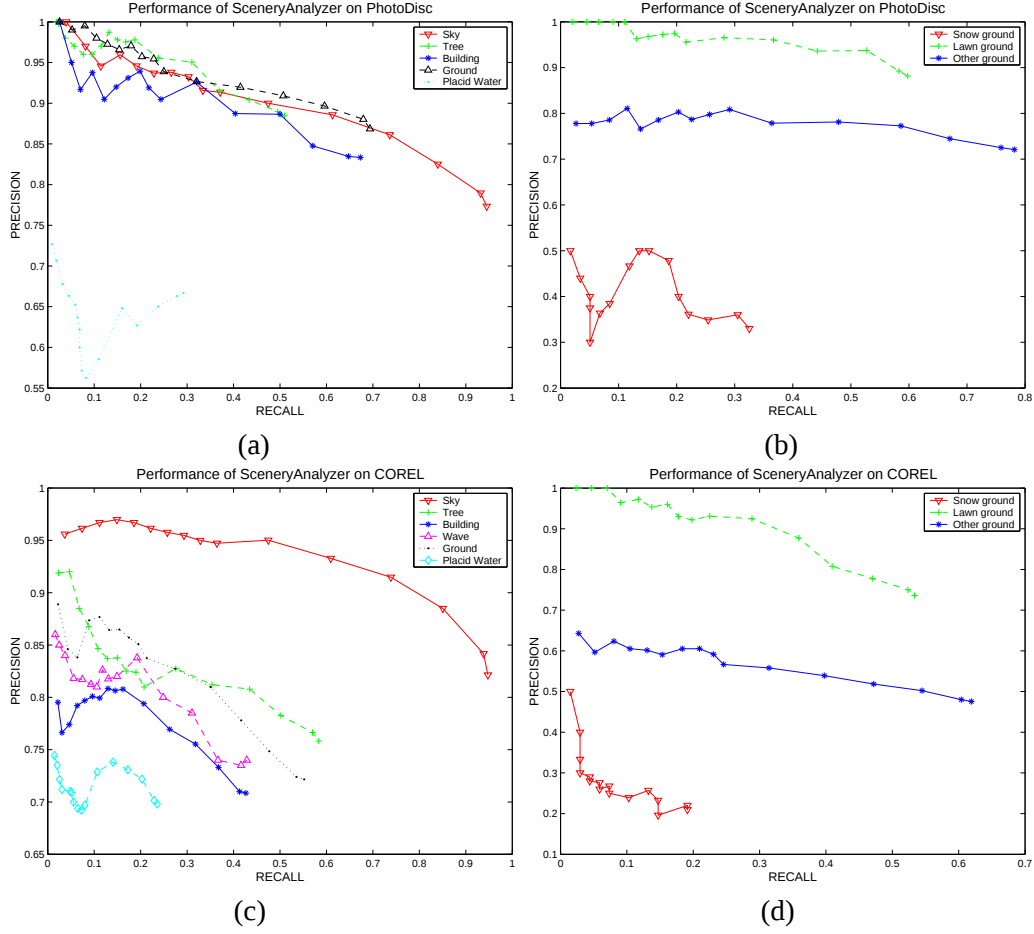


Figure 20: (a) Performance of SceneryAnalyzer using PhotoDisc database with respect to sky, building, tree, placid water, and ground; (b) performance of SceneryAnalyzer using PhotoDisc database with respect to snow, lawn, and other subtypes of ground ; (c) Performance of SceneryAnalyzer using COREL database with respect to sky, building, tree, wave, placid water, and ground; and (d) performance of SceneryAnalyzer using COREL database with respect to snow, lawn, and other subtypes of ground.

Table 2 demonstrates that the SceneryAnalyzer system is effective for the feature combinations listed. In contrast, Table 3 shows that traditional CBIR techniques are not satisfactory when handling combinations of semantic features. SceneryAnalyzer thus provides a new, semantics-based approach to handle queries for images.

Figures 21 and 22 provide sample images and their copies as annotated by SceneryAnalyzer.

<i>features</i>	sky tree	sky wave	tree building	tree ground	sky building lawn	sky tree ground	sky tree building lawn	tree water	sky snow
$ \mathbf{F} $	987	92	571	458	100	262	43	363	78
$ \mathbf{R} $	20	20	20	20	20	20	5	20	20
$ \mathbf{F} \cap \mathbf{R} $	17	17	18	18	18	18	5	11	8
<i>precision</i>	85%	85%	90%	90%	90%	90%	100%	55%	40%
<i>recall</i>	1.7%	18%	3.1%	3.9%	18%	6.9%	12%	3%	10%

Table 2: Performance of SceneryAnalyzer on combinations of features.

<i>features</i>	sky tree	sky wave	tree building	tree ground	sky building lawn	sky tree ground	sky tree building lawn	tree water	sky snow
$ \mathbf{F} $	987	92	571	458	100	262	43	363	78
$ \mathbf{R}_{kb} $	20	20	20	20	20	20	20	20	20
$ \mathbf{F} \cap \mathbf{R}_{kb} $	7	5	8	3	5	3	0	3	1
<i>precision</i>	35%	25%	40%	15%	25%	15%	0%	15%	5%
<i>recall</i>	0.7%	5.4%	1.4%	0.7%	5%	1.1%	0%	0.8%	1.3%
$ \mathbf{R}_{ch} $	20	20	20	20	20	20	20	20	20
$ \mathbf{F} \cap \mathbf{R}_{ch} $	10	2	4	1	3	2	1	5	5
<i>precision</i>	50%	10%	20%	5%	15%	10%	5%	25%	25%
<i>recall</i>	1%	2.2%	0.7%	0.2%	3%	0.8%	2.3%	1.4%	6.4%
$ \mathbf{R}_{ccv} $	20	20	20	20	20	20	20	20	20
$ \mathbf{F} \cap \mathbf{R}_{ccv} $	10	1	8	5	0	2	0	7	0
<i>precision</i>	50%	5%	40%	25%	0%	10%	0%	35%	0%
<i>recall</i>	1%	1.1%	1.4%	1.1%	0%	0.8%	0%	1.9%	0%
$ \mathbf{R}_{daub} $	20	20	20	20	20	20	20	20	20
$ \mathbf{F} \cap \mathbf{R}_{daub} $	7	1	1	2	3	1	1	3	1
<i>precision</i>	35%	5%	5%	10%	15%	5%	5%	15%	5%
<i>recall</i>	0.7%	1.1%	0.2%	0.4%	3%	0.4%	2.3%	0.8%	1.3%
$ \mathbf{R}_{haar} $	20	20	20	20	20	20	20	20	20
$ \mathbf{F} \cap \mathbf{R}_{haar} $	7	3	5	2	1	2	1	7	1
<i>precision</i>	35%	15%	25%	10%	5%	10%	5%	35%	5%
<i>recall</i>	0.7%	3.3%	0.9%	0.4%	1%	0.8%	2.3%	1.9%	1.3%

Table 3: Performance of the five traditional CBIR techniques on combinations of features.



Figure 21: Some examples of scenery images and their annotated copies.



Figure 22: Some examples of scenery images and their annotated copies.

6 Conclusion and discussion

In this paper, we introduced the concept of the monotonic tree as a means to model high-level scenery features. Based on the monotonic tree representation, primitive elements of low-level features such as color, shape, and spatial location can be easily identified, clustered, and combined to form semantically meaningful regions (or features) for images. Thus, images can be automatically annotated with category keywords, including sky, building, tree, wave, lawn, water, snow, and ground. With this annotation, high-level (semantics-based) querying and browsing of images can be supported.

We made simple assumptions about the color, location, harshness, and shape of scenery features. Despite these simple assumptions, the proposed method achieved better results (as measured by precision-recall) than traditional CBIR techniques. These assumptions embody the domain knowledge about scenery images. If we combine the monotonic tree model with more complex domain knowledge, the accuracy of the system in extracting semantic features can be increased.

The reduced monotonic tree captures only the topological structure of the monotonic lines in an image. We may further define a *differential slope* and *differential monotonic tree* to capture differential information. A differential slope may be defined as a sequence of monotonic lines where the gradient is smooth. The differential monotonic tree constructed from these slopes will be capable of incorporating general domain knowledge. Thus, we may use the differential monotonic tree for general semantics extraction.

In future work, we intend to extend the proposed approach to categories other than scenery images. It may be cumbersome to provide a hand-crafted strategy for the identification of structural elements for each semantic feature. It will therefore be necessary for us to use machine-learning techniques to assist in the process of identifying structural elements for general semantic features.

References

- [1] N. Ahuja. Dot pattern processing using voronoi neighborhoods. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 4(3):336–343, 1982.
- [2] N. Ahuja and A. Rosenfeld. Mosaic models for texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):1–11, 1981.
- [3] Narendra Ahuja and Mihran Tuceryan. Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component gestalt. *Computer Vision, Graphics, and Image Processing*, 48(3):304–356, 1989.

- [4] P. Alshuth, T. Hermes, C. Klauck, J. Krey, and M. Roper. Iris - image retrieval for images and videos. In *Proc. of First Int. Workshop of Image Databases and MultiMedia Search, Amsterdam*, pages 170–178, 1996.
- [5] J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Jain, and C.F. Shu. The virage image search engine: An open framework for image management. In *Proceedings of SPIE, Storage and Retrieval for Still Image and Video Databases IV*, pages 76–87, San Jose, CA, USA, February 1996.
- [6] J. Crowley and A. Parker. A representation of shape based on peaks and ridges in the difference of low-pass transform. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 6(2):156–169, March 1984.
- [7] E.R. Dougherty and J.B. Pelz. Texture-based segmentation by morphological granulometrics. In *Advanced Printing of Paper Summaries, Electronic Imaging '89*, volume 1, pages 408–414, Boston, Massachusetts, October 1989.
- [8] Rakesh Dugad and Narendra Ahuja. Unsupervised multidimensional hierarchical clustering. In *IEEE International Conference on Acoustics Speech and Signal Processing, Seattle*, May 1998.
- [9] J.P. Eakins. Automatic image content retrieval - are we getting anywhere. In *Proc. of Third International Conference on Electronic Library and Visual Information Research*, pages 123–135, May 1996.
- [10] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, July 1994.
- [11] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, and B. Dom et al. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):23–32, September 1995.
- [12] D.A. Forsyth, J. Malik, M.M. Fleck, H. Greenspan, T. Leung, S. Belongie, C. Carson, and C. Bregler. Finding pictures of objects in large collections of images. In *Report of the NSF/ARPA Workshop on 3D Object Representation for Computer Vision*, page 335, 1996.
- [13] K. Hirata and T. Kato. Rough sketch-based image information retrieval. *NEC Research & Development*, 34(2):263–273, 1993.
- [14] Berthold Klaus Paul Horn. *Robot Vision*. The MIT Press, forth edition, 1988.
- [15] R.W. Fries J.W. Modestino and A.L. Vickers. Texture discrimination based upon an assumed stochastic texture model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):557–580, 1981.

- [16] Flip Korn, Nikalaos Sidiropoulos, Christous Faloutsos, Eliot Siegel, and Zenon Protopapas. Fast nearest-neighbor search in medical image databases. In *Conference on Very Large Data Bases (VLDB96)*, September 1996.
- [17] Lei Zhu, Aibing Rao and Aidong Zhang. Theory of keyblock-based image retrieval. *ACM Transactions on Information Systems*, Oct 2000.
- [18] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.
- [19] W.Y. Ma and B.S. Manjunath. Netra: A toolbox for navigating large image databases. In *International Conference on Image Processing*, pages I:568–571, 1997.
- [20] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, July 1989.
- [21] B.B. Mandelbrot. *Fractals - Form, Chance, Dimension*. W.H. Freeman, San Francisco, California, 1977.
- [22] B.S. Manjunath and W.Y. Ma. Texture Features for Browsing and Retrieval of Image Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.
- [23] Rajiv Mehrotra and James E. Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, 28(9):57–62, September 1995.
- [24] Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Efficient and Robust Retrieval by Shape Content through Curvature Scale Space. In *Proc. International Workshop on Image Databases and MultiMedia Search*, pages 35–42, Amsterdam, The Netherlands, 1996.
- [25] Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Robust and efficient shape indexing through curvature scale space. In *Proceedings of British Machine Vision Conference*, pages 53–62, Edinburgh, UK, 1996.
- [26] S. Morse. Concepts of use in computer map processing. *Communications of the ACM*, 12(3):147–152, March 1969.
- [27] Greg Pass, Ramin Zabih, and Justin Miller. Comparing images using color coherence vectors. In *Proceedings of ACM Multimedia 96*, pages 65–73, Boston MA USA, 1996.
- [28] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Tools for Content-based Manipulation of Image Databases. In *Proceedings of the SPIE Conference on Storage and Retrieval of Image and Video Databases II*, pages 34–47, 1994.

- [29] Rosalind Picard. A society of models for video and image libraries. Technical Report 360, MIT Media Laboratory Perceptual Computing, 1996.
- [30] C. Robl and G. Farber. Contour tracer for a fast and precise edge-line extraction. In *IAPR Workshop On Machine Vision Applications (MVA98)*, 1998.
- [31] P.L. Rosin and G.A.W. West. Segmentation of edges into lines and arcs. *Image and Vision Computing*, 7(2):109–114, May 1989.
- [32] P.L. Rosin and G.A.W. West. Multi-stage combined ellipse and line detection. In *British Machine Vision Conference (BMVC92)*, pages 197–206, 1992.
- [33] J. Roubal and T.K. Peucker. Automated contour labelling and the contour tree. In *Proc. AUTO-CARTO 7*, pages 472–481, 1985.
- [34] M. Safar, C. Shahabi, and X. Sun. Image retrieval by shape: A comparative study. In *Proceedings of IEEE International Conference on Multimedia and Exposition ICME, USA*, 2000.
- [35] C. Shahabi and M. Safar. Efficient retrieval and spatial querying of 2d objects. In *IEEE International Conference on Multimedia Computing and Systems (ICMCS99)*, pages 611–617, Florence, Italy, June 1999.
- [36] G. Sheikholeslami and A. Zhang. An Approach to Clustering Large Visual Databases Using Wavelet Transform. In *Proceedings of the SPIE Conference on Visual Data Exploration and Analysis IV*, pages 322–333, San Jose, February 1997.
- [37] J. R. Smith and S. Chang. Transform Features For Texture Classification and Discrimination in Large Image Databases. In *Proceedings of the IEEE International Conference on Image Processing*, pages 407–411, 1994.
- [38] John R. Smith and Shih-Fu Chang. VisualSeek: a fully automated content-based image query system. In *Proceedings of ACM Multimedia 96*, pages 87–98, Boston MA USA, 1996.
- [39] J.R. Smith and S.F. Chang. Visualeek: A fully automated content-based image query system. In *ACM Multimedia 96*, 1996.
- [40] Yuqing Song and Aidong Zhang. Monotonic tree. In *The 10th International Conference on Discrete Geometry for Computer Imagery, Bordeaux, France, April 3-5, 2002*. URL=<http://www.cse.buffalo.edu/~ys2/monotree2.pdf>.
- [41] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA, 1996.

- [42] W.D. Stromberg and T.G. Farr. A Fourier-based textural feature extraction procedure. *IEEE Transactions on Geoscience and Remote Sensing*, 24(5):722–732, 1986.
- [43] M.J. Swain and D. Ballard. Color Indexing. *Int Journal of Computer Vision*, 7(1):11–32, 1991.
- [44] T.F. Syeda-Mahmood. Finding shape similarity using a constrained non-rigid transform. In *International Conference on Pattern Recognition*, 1996.
- [45] Y. Tao and W.I. Grosky. Delaunay triangulation for image object indexing: A novel method for shape representation. In *Proceedings of the Seventh SPIE Symposium on Storage and Retrieval for Image and Video Databases*, pages 631–942, San Jose, California, January 1999.
- [46] A.B. Torralba and A. Oliva. Semantic organization of scenes using discriminant structural templates. In *International Conference on Computer Vision (ICCV99)*, pages 1253–1258, 1999.
- [47] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for iso-surface traversal. In *Proc. 13th Ann. Sympos. Comput. Geom.*, pages 212–220, 1997.
- [48] Jia Wang and Raj Acharya. Efficient access to and retrieval from a shape image database. In *IEEE Workshop on Content Based Access of Image and Video Libraries, (CBAIL 98)*, Santa Barbara, June 1998.
- [49] Jia Wang and Raj Acharya. A vertex based shape coding approach for similar shape retrieval. In *ACM symposium on Applied Computing*, pages 520–524, Atlanta, GA, Febraury 1998.
- [50] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. on Computers*, C-20:68–86, January 1971.